La amenaza del Golem

En varias leyendas hebreas de la Edad Media, el golem aparece como un hombre de barro al que un sabio judío insuflaba vida sin ingeniería genética ni nada. Instrucciones de instalación: Ir introduciendo en la boca de la efigie pequeños trozos de papel con las letras que forman uno de los nombres de Dios (yo no intentaría el experimento si no es en compañía de un adulto responsable). El golem solía ser un perfecto sirviente. Su único defecto: ejecutar los comandos de su amo de una forma mecánica y tercamente literal (¿le suena?). Las consecuencias de este comportamiento podían ser cómicas o llevar a un desenlace lleno de hemoglobina, como en el cuento del rabino Ben Bezulel de Praga, relato apropiado para mantener a los peques bien arropados y calladitos en tiempos en que aún no había cine *gore*. El monstruo de Frankenstein viene a ser una encarnación posterior de este mito medieval.

Seguro que quien haya escrito más de cinco líneas de código habrá compartido en algún momento la frustración del rabino de Praga. No basta con dominar el lenguaje que entienden los oídos de barro del golem: hay que saber pensar como él, elaborar modelos para la realidad en los términos de su obtuso y literal entendimiento. Algo más fundamental y de mayor calado que la simple codificación en un lenguaje.

Hasta los usuarios admiten tarde o temprano el hecho de que el ordenador no es más que un golem. No es que el estúpido ordenador haya borrado tu trabajo, guapo, el pobre sólo sigue órdenes. Pero nuestra misión como desarrolladores no es otra que ésta: aislar al usuario de la obstinada inhumanidad de su máquina, hacer de embajadores entre el mundo de la computación y el de los problemas humanos. El usuario está demasiado ocupado para molestarse en pensar como el golem. Nos paga a nosotros para eso. Hasta que nos dimos cuenta, el uso del ordenador para resolver los problemas de la gente corriente estuvo contenido por una barrera de pánico, y con razón: los interfaces de usuario no estaban concebidos a escala humana. Y no se trata de "hacer programas para tontos", expresión utilizada a menudo como única guía de usabilidad. Aunque esta consigna puede ser de ayuda, apesta a displicencia y a torre de marfil. Los que queremos vivir de esto no nos podemos permitir ya esta actitud. De lo que se trata es de darle al programa una metáfora coherente, algo que le presente ante el usuario como una herramienta de este mundo, plegada a las leyes del sentido común. El procesador de textos ofrece una metáfora de máquina de escribir. Esto no ha sido así siempre: ¿Recuerda EDLIN? La hoja de cálculo y sus celdillas, la base de datos y sus fichas, el programa de dibujo y su botecito de pintura, ofrecen otras metáforas olvidadas por lo familiares. Ahora, ordenadores y usuarios esperan que inventemos nuevas metáforas que les ayuden a acercarse. El usuario merece que su programa sea tan sencillo, intuitivo y previsible en su comportamiento como un martillo lo es para el carpintero. Las nuevas e impresionantes prestaciones no añadirán calidad a nuestros programas si no las integramos en la metáfora con accesibilidad y armonía.

Esta tarea exige algo más que simple aptitud para programar: también precisa creatividad. De la misma naturaleza que la que necesita el director de cine para llegar a su público o el publicitario para concebir un mensaje para su campaña. Todo programa se puede mejorar poniendo en juego habilidades creativas. Y no hablo de iconitos horteras y colorines que sólo sirven para despistar. Hablo de aumentar el ancho de banda de la comunicación entre el usuario y su programa, con funcionalidad, con inmediatez, sin estridencias. Para eso sí que hace falta pensamiento creativo y buen gusto. El desarrollador que renuncia a usar la imaginación da validez al tópico que tanto nos perjudica: el de que las máquinas deshumanizan. Para caja tonta, el ordenador. Nuestro oficio consiste en ponerle el duende que le falta, no en contagiarnos de su estupidez.

Por mucho que algún día las máquinas se programen solas, nosotros seguiremos teniendo trabajo. Los únicos desarrolladores condenados a la extinción serán aquéllos que ignoren el antropocentrismo del nuevo software: quienes no sean capaces de desarrollar su cerebro izquierdo para pensar como el golem, y al mismo tiempo potenciar su cerebro derecho para pensar como los usuarios. Este pensamiento doble es el único que puede dar continuidad a nuestro papel de médiums entre el hombre y la máquina.